# Configuration du broker-module-livedata

**Sommaire**



## Description

Le module **broker-module-livedata** permet de mettre à disposition sur le Broker une API HTTP permettant d'accéder aux informations d'un hôte, d'un cluster ou d'un check.

Cette API peut être utilisée pour accéder rapidement aux informations d'éléments supervisés pour ensuite les intégrer dans des outils externes ( *outil de ticketing, récolte de données, etc...* ).

## Activation du module

### Activer le module broker-module-livedata livré par défaut

Par défaut, l'installation ou la mise à jour de Shinken Entreprise va mettre à disposition une définition du module de type "broker_module_livedata" appelé "`broker-module-livedata-example`"

- La configuration de ce module se trouve par défaut dans le fichier : **/etc/shinken/modules/broker-module-livedata.cfg**
- L'activation de ce module s'effectue en ajoutant son nom dans le fichier de configuration du démon **/etc/shinken/brokers/broker-master.cfg** ( *ou le .cfg que vous utilisez pour définir les options de votre broker* ).

  *Exemple :*

  ```
  define broker {
      [...]

      broker_name                                 broker-master

      [...]

      modules                   Module 1, Module 2, Module 3, broker-module-livedata

      [...]
  }
  ```

- Pour prendre en compte le changement de configuration, il faut ensuite redémarrer l'Arbiter :

  ```
  service-shinken-arbiter restart
  ```

## Définir vos propres modules de type broker_module_livedata

Pour pouvoir définir ce module selon vos besoins, il vous sera possible de définir votre module grâce au module d'exemple fourni par défaut.

Pour configurer votre module de type *broker_module_livedata*, commencez par choisir un nom à lui donner.

- Pour l'exemple, nous allons l'appeler "`Mon-Module-Broker-Livedata`".
- Remplacer dans l'exemple le mot "`Mon-Module-Broker-Livedata`" par le nom que vous aurez choisi.

Pour définir votre module à partir du module fourni par défaut, vous devez :

- Copier le fichier de définition du module d'exemple : */etc/shinken-user-example/configuration/daemons/brokers/modules/broker-module-livedata/broker-module-livedata-example.cfg* dans le répertoire de définition des modules **/etc/shinken/modules/** .
  ( *Exemple : /etc/shinken/modules/broker-module-livedata__Mon-Module-Broker-Livedata.cfg* )

```
cp /etc/shinken-user-example/configuration/daemons/brokers/modules/broker-module-livedata/broker-module-livedata-example.cfg /etc/shinken/modules/broker-module-livedata__Mon-Module-Broker-Livedata.cfg
```

- Ouvrez ce fichier ( *broker-module-livedata__Mon-Module-Broker-Livedata.cfg* ) :
  - Modifier la ligne module_name en remplaçant le nom par défaut "**broker-module-livedata**" par le nom que vous avez choisi "`Mon-Module-Broker-Livedata`".

```
...
        #  Module name [ Must be unique
]
[ MANDATORY ]
        #


        module_name                                             Mon-Module-Broker-Livedata
...
```

- Vérifiez que l'utilisateur shinken possède les droits sur ce fichier. Si ce n'est pas le cas, effectuez la commande suivante :

```
chown -R shinken:shinken /etc/shinken/modules/broker-module-livedata__Mon-Module-Broker-Livedata.cfg
```

- Ajoutez le nom du nouveau module au Broker en modifiant le paramètre **modules** du fichier **/etc/shinken/brokers/broker-master.cfg** ( *ou le .cfg que vous utilisez pour définir les options de votre broker* ).

```
define broker {
        [...]

        broker_name                                     broker-master

        [...]

        modules                                             Module 1, Module 2, Module 3, broker-module-livedata

        [...]
}
```

- Redémarrez l'Arbiter pour que le Broker puisse prendre en compte ce nouveau module.

```
service-shinken-arbiter restart
```

## Configuration

La configuration du module se trouve par défaut dans le fichier **/etc/shinken/modules/broker-module-livedata.cfg**

- Vous trouverez aussi systématiquement un exemple dans **/etc/shinken-user-example/configuration/daemons/broker/modules/broker-module-livedata/broker-module-livedata-example.cfg**

### Exemple de fichier de configuration

```
#===============================================================================
# broker-module-livedata
#===============================================================================
# Daemon that can load this module:
# - broker
# This module provides REST API to fetch information on monitored elements
# CFG_FORMAT_VERSION 1
#===============================================================================

define module {

    #  #
    #      MODULE IDENTITY      #
    #  #

    #  Module name [ Must be unique ]                                          [ MANDATORY ]
    #
    module_name                                     broker-module-livedata

    #  Module type [ Do not edit ]                                             [ MANDATORY ]
    #
    module_type                                     broker_module_livedata

    #  #
    #      GENERAL      #
    #  #

    #  API version this module provides
    #
    #           Default : 1 => API_V1
    #           ...     : 2 => API_V2
    #
    broker__module_livedata__api_version                2

    #  Language (only used for the MISSING DATA status)
    #
    #           Default : en => English
    #           ...     : fr => French
    #
    # broker__module_livedata__lang                       en

    #  #
    #      LISTENING PARAMETERS      #
    #  #

    #  IP address to listen to
    #
    #           Default : 0.0.0.0 ( all interfaces )
    #
    # broker__module_livedata__listening_address          0.0.0.0

    #  Port to listen to
    #
    #           Default : 50100
    #
    # broker__module_livedata__listening_port             50100

    #  #
    #      HTTPS PARAMETERS      #
    #  #
```

```
#  Enable this parameter if you want to receive API REST calls in HTTPs mode
#
#           Default : 0 => Disable
#           ...     : 1 => Enable
#
# broker__module_livedata__use_ssl                    0

#  Certificate file
#
#           Default : /etc/shinken/certs/server.cert
#
# broker__module_livedata__ssl_cert                   /etc/shinken/certs/server.cert

#  Key file
#
#           Default : /etc/shinken/certs/server.key
#
# broker__module_livedata__ssl_key                    /etc/shinken/certs/server.key

#  #
#      BROKS GETTER PARAMETERS      #
#  #

#  These parameters allow some internal tuning in broks management in broker-module-livedata

#  Late broks sets catchup
#
#           ...     : 0 => Disable
#           Default : 1 => Enable
#
# broker__module_livedata__broks_getter__activate_late_set_catchup 1

#  Take extra broks sets to manage if more than this parameter sets are waiting
#
#           Default : 10
#
# broker__module_livedata__broks_getter__nb_late_set_allowed_before_catchup 10

#  Stop taking extra broks sets in catchup when we reach this number of broks
#
#           Default : 200000
#
# broker__module_livedata__broks_getter__catchup_broks_managed_by_module_in_a_catchup_loop 200000

#  Continue catchup if too many late broks sets remains after
#
#           ...     : 0 => Disable
#           Default : 1 => Enable
#
# broker__module_livedata__broks_getter__catchup_run_endless_until_nb_late_set_allowed_reached 1

#  Take the lock as soon as getter thread has some broks to manage
#
#           Default : 0 => Disable
#           ...     : 1 => Enable
#
# broker__module_livedata__broks_getter__include_deserialisation_and_catchup_in_lock 0

#  #
#      API V1 PARAMETERS      #
#  #

#  Token used to authenticate API REST calls on this module
#
#           Default : change_me
#
# broker__module_livedata__token                      change_me

#  #
#      API V2 PARAMETERS      #
#  #
```

```
#   Usage Logs   #

#  REST API call can be logged in a specific log file
#
#           Default : 0 => Disable
#           ...     : 1 => Enable
#
# broker__module_livedata__rest_api_log__enable       0

#  File name of the REST API usage logs
#  This file will always be in directory :
#  /var/log/shinken/brokers/modules/broker-module-livedata_API-REST
#  The log file will be daily rotated up to 7 days
#
#           Default : api_v2.log
#
# broker__module_livedata__rest_api_log__file_name    api_v2.log

#  Usage logs verbosity
#
#           Default : NORMAL
#           ...     : VERBOSE
#
# broker__module_livedata__rest_api_log__level        NORMAL

#   Performance Logs   #

#  Log a warning in broker logs when request to api takes longer than this number of seconds
#
#           Default : 3
#
# broker__module_livedata__perf_log__log_call_of_X_seconds_is_an_warning 3

#  Log an error in broker logs when request to api takes longer than this number of seconds
#
#           Default : 6
#
# broker__module_livedata__perf_log__log_call_of_X_seconds_is_an_error 6

#   Statistics   #

#  Maximum time to keep performance statistics, after that time they will be removed
#  All "keeping_last_minutes" parameter will have the value of keeping_last_X_minutes__default,
#  if they are not set
#
#           Default : 60
#
# broker__module_livedata__running_statistics__keeping_last_X_minutes__default 60

#  time to keep performance statistics computed in the running loop of the module
#
#           Default : 60
#
# broker__module_livedata__running_statistics__keeping_last_X_minutes__running_loop 60

#  time to keep performance statistics computed concerning requests
#
#           Default : 60
#
# broker__module_livedata__running_statistics__keeping_last_X_minutes__requests 60

#  The number of requests per second is stored over X hours
#
#           Default : 24
#
# broker__module_livedata__running_statistics__keeping_request_count_per_second_during_X_hours 24

#  #
#      MODULES      #
#  #
```

```
    #  Extra modules can extend scope of broker-module-livedata, adding extra functionalities
    #  this parameter contains coma separated list of submodule names, available submodules are
    #
    #           ...     : livedata-module-event-manager-reader ( allowing to query events from
    #                                                            event container )
    #           ...     : livedata-module-sla-provider ( allowing to query SLA of elements )
    #
    # modules

}
```

## Détails des sections composant le fichier de configuration

### Identification du module

```
...
    #  #
    #      MODULE IDENTITY      #
    #  #

    #  Module name [ Must be unique ]                                                    [ MANDATORY ]
    #
    module_name                                          broker-module-livedata

    #  Module type [ Do not edit ]                                                       [ MANDATORY ]
    #
    module_type                                          broker_module_livedata
...
```

Il est possible de définir plusieurs instances du module *broker_module_livedata*, le paramètre **module_name** permet d'identifier une instance donnée.

| Nom | Type | Unité | Défaut | Commentaire |
|---|---|---|---|---|
| `module_name` | Texte | --- | **broker-module-livedata** | **Nous vous conseillons de choisir un nom en fonction de l'utilisation du module pour que votre configuration soit simple à maintenir.**<br><br>Doit être unique.<br><br>Les caractères autorisés sont les lettres, les chiffres, les caractères _ et - et le nom fourni doit commencer par une lettre |
| `module_type` | Texte | --- | **broker_module_livedata** | Ne peut être modifié. |

### Version de l'API REST ( V1 / V2 )

```
    #  #
    #      GENERAL      #
    #  #
        #  API version this module provides
    #
    #           Default : 1 => API_V1
    #           ...     : 2 => API_V2
    #
    broker__module_livedata__api_version                  2
```

Le module *broker-module-livedata* peut servir une des deux versions d'API disponibles.

Il est possible de choisir cette version via le paramètre **broker__module_livedata__api_version.**

Deux choix sont actuellement disponibles :

- 1 : correspondant à V1 - Les API du broker-module-livedata

- 2 : correspondant à V2 - Les API du broker-module-livedata

| Nom | Type | Unité | Défaut | Commentaire |
|---|---|---|---|---|
| `broker__module_livedata__api_version` | Entier | --- | **1** | Valeurs possibles<br>• **1**<br>• **2** |

### Langue

**/etc/shinken/modules/broker-module-livedata.cfg**

```
#  #
#      GENERAL      #
#  #
#  Language (only used for the MISSING DATA status)
#
#           Default : en => English
#           ...     : fr => French
#
# broker__module_livedata__lang                       en
```

Quand le module *broker-module-livedata* ne reçoit plus de mise à jour pour le statut de certains éléments ( *hôtes, clusters, checks* ), il modifie

- leur statut à ***Données Manquantes*** ,
- leur résultat avec un texte générique expliquant le manque d'information.
  - La langue de ce texte générique est définie par le paramètre **broker__module_livedata__lang**.

| Nom | Type | Unité | Défaut | Commentaire |
|---|---|---|---|---|
| `broker__module_livedata__lang` | Texte | --- | **en** | Les langues disponibles sont<br>• **en** ( *anglais* )<br>• **fr** ( *français* ) |

### Adresse et port d'écoute

```
#  #
#      LISTENING PARAMETERS      #
#  #

#  IP address to listen to
#
#           Default : 0.0.0.0 ( all interfaces )
#
# broker__module_livedata__listening_address          0.0.0.0

#  Port to listen to
#
#           Default : 50100
#
# broker__module_livedata__listening_port             50100
```

Le port et l'adresse d'écoute du module *broker-module-livedata* sont paramétrables via les options suivantes:

| Nom | Type | Unité | Défaut | Commentaire |
|---|---|---|---|---|
| `broker__module_livedata__listening_address` | Texte | Adresse IP | **0.0.0.0** | Adresse IP sur laquelle le module doit se mettre en écoute. |

| | | | | |
|---|---|---|---|---|
| `broker__module_livedata__listening_port` | Entier | Port | **50100** | Port d'écoute du module. |

### Autorisation firewalld

Si **firewalld** est actif sur le système, il faut également autoriser les connexions extérieures vers le port qui vient d'être configuré.

Cela peut être fait en exécutant les commandes suivantes sur le serveur qui fait tourner le Broker (en remplaçant **50100** , ci dessous, par le port configuré dans le fichier de configuration) :

```
firewall-cmd --add-port=50100/tcp
firewall-cmd --runtime-to-permanent
```

### Activation du SSL

```
#  #
#      HTTPS PARAMETERS      #
#  #

#  Enable this parameter if you want to receive API REST calls in HTTPs mode
#
#           Default : 0 => Disable
#           ...     : 1 => Enable
#
# broker__module_livedata__use_ssl                    0

#  Certificate file
#
#           Default : /etc/shinken/certs/server.cert
#
# broker__module_livedata__ssl_cert                   /etc/shinken/certs/server.cert

#  Key file
#
#           Default : /etc/shinken/certs/server.key
#
# broker__module_livedata__ssl_key                    /etc/shinken/certs/server.key
```

Le module *broker-module-livedata* peut, au choix, servir le protocole **http** ou **https** sur le port configuré ci-dessus.

Le mode **https** se configure avec les paramètres suivants :

| Nom | Type | Unité | Défaut | Commentaire |
|---|---|---|---|---|
| `broker__module_livedata__use_ssl` | Booléen | --- | **0** | Paramètre activant le mode SSL ( *https* ).<br>Valeurs:<br>• **0** ( *non* )<br>• **1** ( *oui* )<br>Par défaut le module fonctionne en mode http sur le port configuré ci-dessus. |
| `broker__module_livedata__ssl_cert` | Texte | --- | **/etc/shinken/certs/server.cert** | Chemin du fichier contenant le certificat. |

| | | | | |
|---|---|---|---|---|
| broker__module_livedata__ssl_key | Texte | --- | **/etc/shinken/certs/server.key** | Chemin du fichier contenant la clé du certificat. |

### Absorption des broks ( information de supervision venant des Schedulers )

```
#  #
#      BROKS GETTER PARAMETERS      #
#  #

#  These parameters allow some internal tuning in broks management in broker-module-livedata

#  Late broks sets catchup
#
#           ...     : 0 => Disable
#           Default : 1 => Enable
#
# broker__module_livedata__broks_getter__activate_late_set_catchup 1

#  Take extra broks sets to manage if more than this parameter sets are waiting
#
#           Default : 10
#
# broker__module_livedata__broks_getter__nb_late_set_allowed_before_catchup 10

#  Stop taking extra broks sets in catchup when we reach this number of broks
#
#           Default : 200000
#
# broker__module_livedata__broks_getter__catchup_broks_managed_by_module_in_a_catchup_loop 200000

#  Continue catchup if too much late broks sets remains after
#
#           ...     : 0 => Disable
#           Default : 1 => Enable
#
# broker__module_livedata__broks_getter__catchup_run_endless_until_nb_late_set_allowed_reached 1

#  Take the lock as soon as getter thread has some broks to manage
#
#           Default : 0 => Disable
#           ...     : 1 => Enable
#
# broker__module_livedata__broks_getter__include_deserialisation_and_catchup_in_lock 0
```

Ces paramètres sont dédiés au fonctionnement interne au module, il est fortement recommandé de ne pas les modifier sans votre support dédié.

Le fonctionnement du thread de récupération des **broks** peut être configuré via certains paramètres, afin de modifier son "agressivité".

Pendant la mise à jour des données de supervision, le module ne peut pas répondre aux requêtes HTTP qu'il reçoit.

Principe de l'algorithme d'absorption des broks :

1. Attente de broks à traiter
2. Récupération de broks en retard ( *fonctionnalité de rattrapage* )
3. Désérialisation des broks
4. Entrée en session critique ( *les requêtes à l'API sont bloquées* )
5. Traitement des broks
6. Libérer la session critique et attendre de nouveaux broks, **ou** continuer l'absorption de broks ( *en cas de retard important, on repart à l'étape 1. en restant sur la session critique* )

| Nom | Type | Unité | Défaut | Commentaire |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| `broker__module_livedata__broks_getter__activate_late_set_catchup` | Booléen | --- | **1** | Utilisation de la fonctionnalité de rattrapage pour absorber des **broks** en retard :<br>• **1** : Activé<br>• **0** : Désactivé |
| `broker__module_livedata__broks_getter__nb_late_set_allowed_before_catchup` | Nombre | Nombre de broks set | **10** | Nombre de **brok set** en attente toléré.<br>Au-dessus de ce nombre, les **brok set** sont immédiatement récupérés par l'algorithme de rattrapage pour être traités immédiatement. |
| `broker__module_livedata__broks_getter__catchup_broks_managed_by_module_in_a_catchup_loop` | Nombre | Nombre de broks | **200000** | Nombre maximal de **broks** que l'algorithme de rattrapage récupère avant de lancer le traitement.<br>Ce paramètre permet de borner la consommation mémoire et le temps d'exécution d'un tour de boucle de traitement. |
| `broker__module_livedata__broks_getter__catchup_run_endless_until_nb_late_set_allowed_reached` | Booléen | --- | **1** | Après traitement des **broks**, si le nombre de **brok set** en retard est trop élevé,<br>• **1** : continuer le rattrapage et absorber des **broks** en retard en restant sur la session critique ( *"avec le lock"* )<br>• **0** : arrêter l'absorption de **brok** et libérer la session critique ( *rendre le lock* ) |
| `broker__module_livedata__broks_getter__include_deserialisation_and_catchup_in_lock` | Booléen | --- | **0** | Dans le cas où vous voulez disposer d'un maximum de temps CPU pour traiter les **broks** en retard, vous pouvez activer ce paramètre afin de bloquer les requêtes à l'API dès la phase 2 ( *Récupération de broks en retard* ) puis une fois les broks rattrapés passés en Phase 5 ( *Traitement des broks* ).<br>Deux valeurs possibles pour ce paramètre :<br>• **1** : Activé<br>• **0** : Désactivé |

### Paramètres spécifiques de l'API V1

#### Configuration de l'authentification

```
#  #
#      API V1 PARAMETERS      #
#  #

#  Token used to authenticate API REST calls on this module
#
#           Default : change_me
#
# broker__module_livedata__token                      change_me
```

Afin de sécuriser l'utilisation de l'API du module, un token est nécessaire pour chaque requête. Ce token peut être défini dans la configuration en modifiant le paramètre **broker__module_livedata__token.**

| Nom | Type | Unité | Défaut | Commentaire |
|---|---|---|---|---|
| `broker__module_livedata__token` | Texte | --- | **change_me** | Chaîne de texte utilisée pour chaque requête au module * |

*** Avertissement**

Cette valeur est utilisée en paramètre d'URL de type ***GET*** . Si elle contient des caractères interdits dans une URL, ils devront être échappés (URL encodés) lors de son utilisation pour une requête

| caractère interdit | : | / | ? | # | [ | ] | @ | ! | $ | & | ' | ( | ) | * | + | , | ; | = | % | (espace) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| remplacement | %3A | %2F | %3F | %23 | %5B | %5D | %40 | %21 | %24 | %26 | %27 | %28 | %29 | %2A | %2B | %2C | %3B | %3D | %25 | %20 ou + |

Pour plus d'information https://developer.mozilla.org/fr/docs/Glossary/percent-encoding et rfc3986

Exemple: pour le token ***ch@nge_me*** il faudra utiliser l'url **http://broker-module-livedata:50100/api/v1/all-monitored-elements?token=ch%40nge_me**

### Paramètres spécifiques de l'API V2

L'API REST V2 du module *broker-module-livedata* permet, optionnellement, d'enregistrer toutes les requêtes qui sont faites via des **logs d'utilisation**.

De plus, pour suivre le bon fonctionnement du module, des **logs de performance** permettent de tracer les requêtes lentes.

#### Configuration des logs d'utilisation

```
#  #
#      API V2 PARAMETERS      #
#  #

#   Usage Logs

#  REST API call can be logged in a specific log file
#
#           Default : 0 => Disable
#           ...     : 1 => Enable
#
# broker__module_livedata__rest_api_log__enable        0

#  File name of the REST API usage logs
#  This file will always be in directory :
#  /var/log/shinken/brokers/modules/broker-module-livedata_API-REST
#  The log file will be daily rotated up to 7 days
#
#           Default : api_v2.log
#
# broker__module_livedata__rest_api_log__file_name     api_v2.log

#  Usage logs verbosity
#
#           Default : NORMAL
#           ...     : VERBOSE
#
# broker__module_livedata__rest_api_log__level         NORMAL
```

La fonctionnalité de **logs d'utilisation** est définie plus en détails sur la Broker - Les logs du module broker-module-livedata.

Ces logs sont enregistrés dans un fichier spécifique, les paramètres de configuration sont les suivants :

| Nom | Type | Unité | Défaut | Commentaire |
|---|---|---|---|---|
| `broker__module_livedata__rest_api_log__enable` | Booléen | --- | **0** | Par défaut, les logs d'utilisation sont désactivés.<br>Valeurs possibles :<br>• **0**<br>• **1** |

| | | | | |
|---|---|---|---|---|
| `broker__module_livedata__rest_api_log__file_name` | Texte | --- | **api_v2.log** | Nom du fichier qui stockera les logs d'utilisation. Ce fichier sera dans le dossier */var/log/shinken/brokers/modules/broker-module-livedata_API-REST*<br><br>Le journal fera l'objet d'une rotation quotidienne jusqu'à 7 jours. |
| `broker__module_livedata__rest_api_log__level` | Texte | --- | **NORMAL** | Valeurs possibles :<br><br>• **NORMAL** pour des logs minimaux.<br>• **VERBOSE** pour des logs détaillés. |

### Logs de performance

```
#  #
#      API V2 PARAMETERS      #
#  #

#   Performance Logs

#  Log a warning in broker logs when request to api takes longer than this number of seconds
#
#           Default : 3
#
# broker__module_livedata__perf_log__log_call_of_X_seconds_is_an_warning 3

#  Log an error in broker logs when request to api takes longer than this number of seconds
#
#           Default : 6
#
# broker__module_livedata__perf_log__log_call_of_X_seconds_is_an_error 6
```

Les logs de performance sont enregistrés dans le fichier de logs du Broker.

Leur format est détaillé sur la Broker - Les logs du module broker-module-livedata.

Les paramètres suivants permettent d'influer sur leur génération :

| Nom | Type | Unité | Défaut | Commentaire |
|---|---|---|---|---|
| `broker__module_livedata__perf_log__log_call_of_X_seconds_is_an_warning` | Entier | secondes | **3** | Si le temps d'exécution d'une requête dépasse cette durée, un log de niveau **WARNING** sera généré dans le fichier des logs du Broker |
| `broker__module_livedata__perf_log__log_call_of_X_seconds_is_an_error` | Entier | secondes | **6** | Si le temps d'exécution d'une requête dépasse cette durée, un log de niveau **ERROR** sera généré dans le fichier des logs du Broker |

### Gestion de statistiques pour la supervision du module

```
#  #
#      API V2 PARAMETERS      #
#  #

#   Statistics

#  Maximum time to keep performance statistics, after that time they will be removed
#  All "keeping_last_minutes" parameter will have the value of keeping_last_X_minutes__default,
#  if they are not set
#
#           Default : 60
#
# broker__module_livedata__running_statistics__keeping_last_X_minutes__default 60

#  time to keep performance statistics computed in the running loop of the module
#
#           Default : 60
#
# broker__module_livedata__running_statistics__keeping_last_X_minutes__running_loop 60

#  time to keep performance statistics computed concerning requests
#
#           Default : 60
#
# broker__module_livedata__running_statistics__keeping_last_X_minutes__requests 60

#  The number of requests per second is stored over X hours
#
#           Default : 24
#
# broker__module_livedata__running_statistics__keeping_request_count_per_second_during_X_hours 24
```

Pour limiter la quantité de données à garder, les paramètres suivants sont disponibles :

| Nom | Type | Unité | Défaut | Commentaire |
|---|---|---|---|---|
| `broker__module_livedata__running_statistics__keeping_last_X_minutes__default` | Entier | minutes | **60** | Valeur par défaut pour les deux paramètres suivants, si vous ne le définissez pas explicitement. |
| `broker__module_livedata__running_statistics__keeping_last_X_minutes__running_loop` | Entier | minutes | **60** | Intervalle de temps sur lequel on conserve les mesures de performance de la boucle principale du module. |
| `broker__module_livedata__running_statistics__keeping_last_X_minutes__requests` | Entier | minutes | **60** | Intervalle de temps sur lequel on conserve les mesures de temps des requêtes traitées. |
| `broker__module_livedata__running_statistics__keeping_request_count_per_second_during_X_hours` | Entier | heures | **24** | Nombre d'heures pendant lesquelles on conserve le nombre de requêtes par secondes. |

### Sous modules

```
#  #
#      MODULES      #
#  #

#  Extra modules can extend scope of broker-module-livedata, adding extra functionalities
#  this parameter contains coma separated list of submodule names, available submodules are
#
#           ...     : livedata-module-event-manager-reader ( allowing to query events from
#                                                            event container )
#           ...     : livedata-module-sla-provider ( allowing to query SLA of elements )
#
# modules
```

L'API REST V2 peut être enrichie via l'utilisation des sous modules du *broker-module-livedata*

Pour activer un sous module, il faut ajouter son nom dans le paramètre **modules** :

| Nom | Type | Unité | Défaut | Commentaire |
|---|---|---|---|---|
| `modules` | Texte | --- | | Liste de noms de sous modules, séparés par des virgules. Par défaut, aucun sous module n'est activé |

Voici la liste des modules actuellement disponibles, pouvant s'attacher au module broker-module-livedata ( voir Module broker-module-livedata )

| Type de module | Configuration | Documentation | Description |
|---|---|---|---|
| `livedata-module-sla-provider` | Module livedata-module-sla-provider | V2 - ( READ ) /api/v2/sla -- OPTIONNEL -- | Ajoute la possibilité de récupérer les données SLAs. |

# Vérification du bon fonctionnement ( est ce que vous avez accès aux APIs )

## Tester l'API V1

Depuis une des machines qui va faire les requêtes aux API,  faire le test suivant :

```
curl  -s -S -k http://machinedubroker:50100/api/v1/ping
```

| Résultat attendu |
|---|
| `{"response": "pong"}` |

Si la réponse attendue n'arrive pas, contrôler les autorisation d'accès du firewall

## Tester l'API V2

Depuis une des machines qui va faire les requêtes aux API,  faire le test suivant :

```
curl  -s -S -k http://machinedubroker:50100/api/v2/ping
```

**Résultat attendu**

```
{"response": "pong"}
```

Si la réponse attendue n'arrive pas, contrôler les autorisation d'accès du firewall

## Vérification du firewall

### Si vous avez firewalld ( firewall par défaut de la Redhat )

Si **firewalld**  est actif sur le système, il faut également autoriser les connexions extérieures vers le port qui vient d'être configuré.

Sur la machine du Broker qui fait tourner le module livedata, vérifiez que le port est ouvert dans votre firewall :

```
firewall-cmd --list-ports
```

**Exemple de résultat**

```
80/tcp 7763/tcp 7765/tcp 7766/tcp 7767/tcp 7768/tcp 7769/tcp 7770/tcp 7771/tcp 7772/tcp 7773/tcp 7777/tcp
7780/tcp 50000/tcp
```

Dans cet exemple, le port 50100/tcp n'est pas listé, il est donc bloqué par défaut.

Il faut modifier le firewall pour autoriser les connexions.

Cela peut être fait en exécutant les commandes suivantes sur le serveur qui fait tourner le Broker ( *en remplaçant **50100**, ci dessous, par le port configuré dans le fichier de configuration, si vous n'utilisez pas celui par défaut* ) :

```
firewall-cmd --add-port=50100/tcp
firewall-cmd --runtime-to-permanent
```